2009-IOT-4 (27)
2009 / 3 / 5

認証デバイスを用いたOSの起動・終了制御システムにおける起動時間の短縮

高田 真吾† 佐藤 聡† 新城 靖† 中井 央†† 板野 肯三†

† 筑波大学 大学院 システム情報工学研究科 〒305–8573 茨城県つくば市天王台 1-1-1
†† 筑波大学 大学院 図書館情報メディア研究科 〒305–8573 茨城県つくば市天王台 1-1-1
E-mail: {takada@softlab.cs, akira@cc, yas@is, nakai@slis, k3itano@cs}.tsukuba.ac.jp

あらまし 我々は認証デバイスを用いることにより利用者とハードウェアを識別し，適切な計算機環境を提供する仕組みとして，OS 制御レイヤーを利用した仕組みを提案・実装している．従来の実装では，利用者が利用する OS のディスクイメージ全体を事前にダウンロードし，仮想計算機のゲスト OS の形で利用者に提供している．そのため，利用者がその OS を利用可能になるまでにかかる時間は，ディスクイメージサイズに比例して増加する．本論文では，ランダムアクセス可能なネットワークファイルシステムを利用することにより，利用者から見た OS の起動時間の短縮化を行うことについて述べる．これにより，ゲスト OS として実用上最小構成の Windows XP を利用する際の起動所要時間を 75%削減した．

キーワード 起動終了制御，認証デバイス，電子証明書，仮想計算機

Shortening Boot Times of Guest OSes on OS Boot/Shutdown Control System using Authentication Devices

Shingo TAKADA†, Akira SATO†, Yasushi SHINJO†, Hisashi NAKAI††, and Kozo ITANO†

† Graduate School of Systems and Information Engineering, University of Tsukuba
Tennoudai 1–1–1, Tsukuba-shi, 305–8573 Japan
†† Graduate School of Library, Information and Media Studies, University of Tsukuba
Tennoudai 1–1–1, Tsukuba-shi, 305–8573 Japan
E-mail: {takada@softlab.cs, akira@cc, yas@is, nakai@slis, k3itano@cs}.tsukuba.ac.jp

Abstract We are developing an OS boot/shutdown control system which provides a suitable computer environment by identifying both an user and a computer using authentication devices. In the previous system, since the computer environment is provided as guest OS disk images running on a virtual machine. and we have to download entire diskimage files in advance, the time for booting a suitable OS is in proportion to its file size. In this paper, we propose a shortening method which can boot a suitable OS more quickly. This method uses a random-accessible network file system to fetch OS images. Using this method, the time for booting a minimal image of Windows XP was shortened by 75%.

Key words Boot/Shutdown Control, Authentication Device, Electric Certification, Virtual Machine

1. はじめに

企業や大学など，多数のコンピュータを導入している組織において，利用者とハードウェアを識別し，その組み合わせに応じて許可された複数の計算機環境を提供できる仕組みがあれば，組織における計算機利用の柔軟性は向上する．この仕組みを実現するため，我々は認証デバイスを用いて OS の起動・終了を制御する手法を提案し，実装している [1], [2]．従来手法では，利用者が実際に利用する OS(以下**利用者用 OS**) を起動するために配信サーバから OS イメージをあらかじめダウンロードしておき，完了後にその OS を起動していた．そのため，利用者用 OS を利用できるようになるまでにかかる時間は，OS イメージのサイズに比例してしまうという問題があった．

本稿では，従来手法の問題点である利用者用 OS の起動まで

の所要時間を短縮するために，ランダムアクセス可能なファイルシステムを用いる手法を提案する．

2. 関 連 研 究

TPM のような，チップに格納されたハードウェア固有情報を利用することで実行されているハードウェアを識別でき，かつネットワーク上のサーバから OS を起動できるような仕組みとして，Trusted HTTP-FUSE KNOPPIX [3] がある．

HTTP-FUSE KNOPPIX では，KNOPPIX の起動に必要なファイルを HTTP を使って取得する．その際に，取得対象とする KNOPPIX のバリエーションを選択できるため，教育用や数値計算用など，用途に応じた OS 環境をネットワーク経由で起動することができる．必要な部分を含むブロックの形でランダムアクセスできるほか，一度取得したブロックはクライアント側にキャッシュされるため，2 回目以降は高速に動作する．

提案システムでは，ディスクイメージへのアクセスに Kerberos 化された NFS を用いる．NFS を用いるため，必要な部分へのランダムアクセスが可能である．HTTP-FUSE KNOPPIX では，用途に応じた KNOPPIX の環境を提供することができるが，提案システムでは提供可能な計算機環境を KNOPPIX(Linux) に限定しない．OS 制御レイヤーで用いる仮想計算機モニタで，ゲスト OS として起動することが可能な OS であれば対応可能である．

3. 提 案 方 式

我々は，認可された利用者の持つ認証デバイスと，認可されたハードウェアの組み合わせに対して，その組み合わせについて許可された OS を利用できるようなシステムの実現を目指している．

このシステムは，以下に示すことを目標としている．

（1） 認証デバイスとハードウェアの組み合わせから, 起動が許可される OS が決定されること

（2） システムの利用が許可されていないハードウェアからのディスクイメージに対するアクセスを防ぐことができること

（3） 認証デバイスやハードウェア，OS の種類が増えても，十分なサービス品質を保証できるだけのスケーラビリティをもつこと

我々は，これらの目標を達成するシステムの設計および実装を行った [1], [2]．しかし，この従来手法では，提供する OS のディスクイメージサイズが大きくなると起動所要時間が長くなってしまうという問題があった．そこで，本研究では上記の目標を達成したまま OS の起動所要時間を短縮する手法を提案する．

ハードウェアの識別には，安全性の観点から TPM [4] のように複製が極めて難しいモジュールを用いるべきである．しかし，プロトタイプの作成や検証を行う上で TPM の利用が困難であるため，本研究では同等の処理をソフトウェアで行えるよう，X.509 形式の証明書を用いる．また提案手法では，ランダムアクセス可能なファイルシステムとして Kerberos 化された NFS [5] を利用するが，Kerberos 化された NFS では TPM や X.509 形式の証明書を直接扱えないため，その橋渡しを行う仕組みが必要となる．そこで，証明書を使った認証により Kerberos のプリンシパルを取得する仕組みも提案する．

なお，提案手法においては，利用者が操作する端末について，次の要素を TCB(Trusted Computing Base) として扱う．

- 仮想計算機のホスト OS のカーネル
- OS 制御レイヤーとそのサブシステム
- ホスト OS 上で動作する root 特権を持つサービス

これらの要素は常に信頼できる (trusted) とする．また，物理的なセキュリティ脅威も起こらないとする．つまり，ホスト OS のカーネルが想定外の動作を行ったり，OS 制御レイヤーが想定外の振る舞いをしたりすること，端末の HDD に直接アクセスが行われ構成ファイルが盗まれたりすることなどは考えない．

これは，将来的にはこの提案手法がファームウェアの形で実装されることを想定してのものである．そのため，提案手法のシステムを組み込んだハードウェアに固有の情報を持たせるのではなく，外部の (trusted な) ハードウェアモジュールにハードウェア固有の情報を持たせることで動作するよう設計した．これにより，システム部と固有情報部を切り分けることが可能となり，ファームウェア化する際に都合が良くなる．

3.1 従来手法の問題点と解決方法

Windows のような GUI を持つ OS では，そのサイズは一般的に数 GB となる．例として，Windows XP Professional ServicePack3 をインストールし，アンチウイルスソフトウェア (Symantec AntiVirus 10) をインストールした上で Windows Update を行った場合では，システムドライブの総使用量はおよそ 3.5GB であった．

従来手法では，OS イメージの転送に HTTPS を用いており，その転送速度は約 30MBps であった．そのため，3.5GB のディスクイメージを転送するのにおよそ 2 分かかる．ほぼ最小構成での Windows の場合でもこのサイズとなるため，利用者の使いたいアプリケーションがインストールされていればさらに所要時間は延びてしまう．

従来手法と提案手法との比較を図 1 に示す．提案手法では，OS イメージのダウンロードにかかる時間のかわりに，Kerberos 化された NFS サーバのファイルを利用できるようマウント処理を行う時間が必要となる．この時間は，イメージ全体をダウンロードする時間に比べて非常に短く，また対象となる OS イメージのファイルサイズによらないという特徴があるため，従来手法に比べ，起動までの所要時間を短縮できると考えられる．

3.2 ネットワークファイルシステムの要件

提案手法では，利用者用 OS のディスクイメージの配信にネットワークを経由するファイルシステムとして次のような要件を満たすファイルシステムが必要となる．

（1） 権限を持たないアクセスを排除できること

（2） ランダムアクセス可能であること

(1) はシステムの安全性や完全性の保証のために必要となる．これが満たされない場合，本来であればその OS を起動できないはずの利用者が OS を利用できるようになってしまう．(2) は，本研究で提案する起動所要時間の短縮化手法において必須

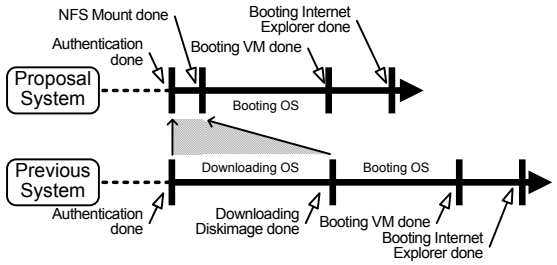

図 1　起動処理の比較

Fig. 1　Comparing boot sequence

となる要件である．

これらの要件を満たす方法の一つとして，Kerberos 化された NFS(Kerberized NFS) がある．Kerberos 化された NFS では，Kerberos のプリンシパルに基づいてアクセス制御が行われるため，適切な権限を持たないアクセスを排除することが可能であるうえ，相互認証により通信相手の検証が可能であるため，なりすましを防ぐこともできる．また，ランダムアクセス可能という特徴も持つため，提案手法で要求される条件を全て満たしている．

この要件を満たすようなほかの手法としては，NFS を VPN 上で利用するという手法も考えられる．NFS だけでは (1),(2) の双方を同時に満たす事ができない．そこを VPN の認証処理で補う手法である．しかし，この手法ではクライアントは VPN サーバを経由して NFS サーバにアクセスするため，VPN サーバに通信が集中してしまうことが考えられる．Kerberos 化された NFS では，クライアントとサーバが直接通信を行うため，このように通信が集中することがない．また，クライアントの増加に伴ってサーバシステムを増強する場合には，単に Kerberos 化された NFS サーバを追加するだけとなるため，容易に増強可能である．これらの理由から，本研究では Kerberos 化された NFS を採用する．

3.3　システム概要

本研究で提案するシステムの概要を図 2 に示す．提案システムは，利用者が実際に操作するハードウェアである**クライアントシステム**と，クライアントシステムに OS の選択肢や OS 環境を提供する**サーバシステム**からなる．図中において，クライアントシステム以外のサーバは全てサーバシステムの構成要素である．

クライアントシステムは，認証デバイスの制御や監視，利用者用 OS の起動・終了制御などを行う．

サーバシステムは，Kerberos Keytab 配信サーバとユーザエントランスサーバ，Kerberos KDC(Key Distribution Center) と Kerberos 化された NFS サーバ群，そしてそれらが利用するバックエンドデータベースから構成される．

Kerberos Keytab 配信サーバは，クライアントシステムが Kerberos のサービスを利用するためのプリンシパルに必要な鍵情報ファイル (keytab ファイル) を配信するためのサーバである．必要となるプリンシパルは，ホスト自身を示す `host/hostname@REALM` と NFS クライアントとして動作する際

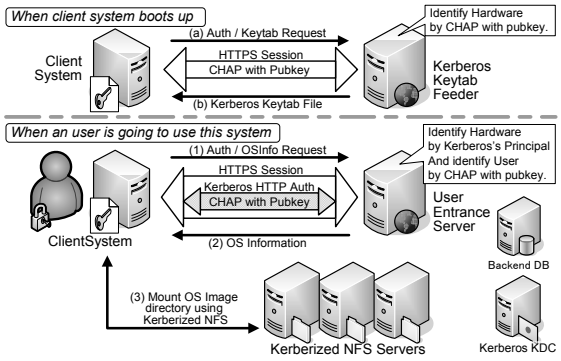

図 2　提案システム概要

Fig. 2　Overview of proposal system

に必要となる `nfs/hostname@REALM` である．クライアントシステムは，ローカルファイルに保存されている X.509 証明書を用いてクライアントシステム自身の起動時に一度だけ認証を行い，これらのプリンシパルを利用する際に必要となる keytab ファイルを取得する．

ユーザエントランスサーバは，システムを利用しようとしている利用者を認証し，起動が許可されている OS の情報を提供するサーバである．利用者の持つ認証デバイスと，ハードウェアが持つ Kerberos プリンシパルから利用者とハードウェアを識別し，その組み合わせで起動が許可されている OS の情報をバックエンドデータベースから取得し，クライアントシステムに返す．OS の情報とは，OS の名前やその OS イメージを担当する NFS サーバのアドレス，マウント時に必要となるパス名や，OS に関する説明文などの情報である．クライアントシステムは，その情報をもとに利用者用 OS を起動し，その環境を利用者に提供する．

利用者が利用する OS のディスクイメージは，Kerberos 化された NFS サーバ群によりクライアントシステムに提供される．クライアントシステムはユーザエントランスサーバから取得した NFS サーバに接続を行うため，負荷の状態にあわせて NFS サーバの振り分けを行うことにより，容易に負荷分散を実現可能であり，スケーラビリティが確保される．

Kerberos KDC は，Kerberos を利用する際に必要となるサーバである．チケットを発行するためのチケットである TGT(Ticket Granting Ticket) の発行や，サービスを利用するためのチケットであるサービスチケットの発行などを行う．

バックエンドデータベースは，上記のサービスを提供する上で必要となる様々な情報を格納する．具体的には，利用者やハードウェアの情報や，どの組み合わせでどの OS が起動できるかを記したアクセス制御リストなどが格納される．

3.4　クライアントシステムにおける OS の起動・終了制御

クライアントシステムにおける OS の起動・終了制御の実現には，従来手法 [2] を用いる．これは，HTTPS 通信を用いて利用する OS のディスクイメージをダウンロードし，仮想計算機モニタ Xen [6] を用いてその OS を起動するという手法である．

Xen では，起動する OS のディスクイメージをファイルとして取り扱う．従来手法では，このディスクイメージをあらかじ

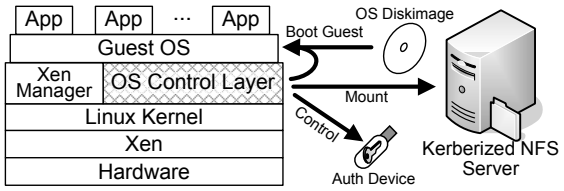

図 3　クライアントシステムの構成
Fig. 3　Overview of ClientSystem

めクライアントシステムのローカルディスクにファイルとしてダウンロードしていた．そのため，OS の起動に要する時間にはダウンロードに要する時間が含まれていた．

提案手法では，このディスクイメージをネットワークファイルシステム上のファイルとする．つまり，OS の起動中にディスクイメージの必要な箇所にネットワークを介してアクセスし，OS の起動にかかる処理とネットワークアクセスにかかる処理とが並行に行われるために，OS の起動に要する時間が短縮される．提案手法の概念図を図 3 に示す．

3.5　利用者・ハードウェアの識別

提案手法では，利用者とハードウェアの双方を識別できる必要がある．今回は，利用者の識別には利用者の持つ認証デバイスを用いる．クライアントシステムのハードウェアの識別には，前述の通りハードウェアごとに固有な X.509 証明書を持たせ，その証明書を用いて識別を行う．

まず，クライアントシステムの識別には Kerberos のプリンシパルを用いる．クライアントシステム自身の起動処理において，Kerberos Keytab 配信サーバとの間で自らが持つ固有な証明書を用いて認証を行う．認証に成功すると，配信サーバは Kerberos プリンシパルの鍵である keytab ファイルを返す．クライアントシステムはこの鍵を使い，動作に必要なチケットを取得する．クライアントシステム自身がシャットダウンされるまでの間，このプリンシパルによりクライアントシステムの識別を行う．

次に，利用者の識別には利用者の持つ認証デバイスに格納された証明書を用いる．利用者の認証は，ユーザエントランスサーバが行う．ユーザエントランスサーバとの通信には HTTPS を用い，かつその上で Kerberos プリンシパルによる認証を行う．これにより，HTTPS による安全な通信路を用いることができる上に，サーバは認証リクエストがどのクライアントシステムから行われているかを認識することができる．そのセッションを用いて，利用者の認証デバイスを用いて認証を行う．これにより，どのクライアントシステムでどの利用者が認証処理を行っているかを認識できるようになる (図 4)．

3.6　アクセス制御ポリシーの記述

提案手法では，利用者とハードウェアの組について起動可能な OS を設定できる仕組みを提供する．しかし，これをアクセス制御リストの形で記述した場合，利用者数 × ハードウェア数 個のルールが必要となり，莫大な量となってしまう．

そこで，本研究ではアクセス制御を行う場合には人やハードウェアのグループに対してルールを決めることが多いことを利用し，本研究ではアクセス制御の粒度を個々の利用者やハードウェアごとではなく，グループ単位とする．つまり，利用者のグループ，ハードウェアのグループ，OS の組についてルールを記述することを提案する．

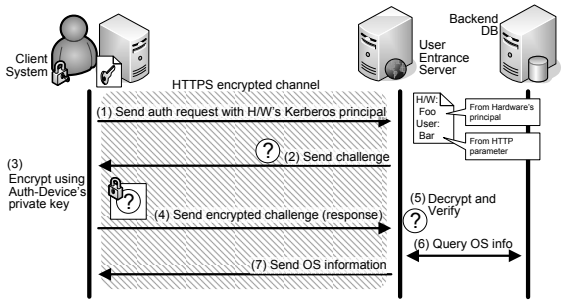

図 4　認証処理の流れ
Fig. 4　Authentication flow

アクセス制御のルールは，次のように記述する．

```
allow_os --UG Lab_Admin --HG ALL \
         --OS Linux_for_Admin
```

この記述では，`Lab_Admin` グループに所属する人が，**全ての**グループに所属するハードウェアを利用する際，`Linux_for_Admin` という OS を利用できることを意味する．

このグルーピングにより，アクセス制御リストの規模が拡大する速度を抑えられるほか，利用者やハードウェアを新たに追加する場合にも，新しいルールを書くのではなく追加した要素を適切なグループに配置するだけで済むため，利用者やハードウェアをシステムに追加する際のコストを軽減することもできる．

4.　実　　装

3 章で述べた提案方式を実際に作成した．Kerberos KDC と Kerberos 化された NFS サーバに関しては，それらを改変することなく利用した．

4.1　Kerberos Keytab 配信サーバ

Kerberos Keytab 配信サーバでは，3 章で示したような認証処理を行う．この処理を Apache 上で動作する CGI により実装した．この CGI は，認証要求を受けチャレンジデータを送出する受付スクリプトと，クライアントからのレスポンスを受け検証・Kerberos keytab ファイルの提供を行う応答スクリプトの 2 つで構成する．どちらも HTTPS サーバ認証セッション上でのやりとりを行う．

受付スクリプトは，HTTP のパラメータによりハードウェア名を受け取る．そのハードウェア名について固有なデータをランダムに生成し，これをチャレンジデータとする．このデータをクライアントシステムに返送し，また応答スクリプトで検証する際のためにサーバ上に保存しておく．

応答スクリプトは，同様にハードウェア名を受け取るほか，クライアントシステムが送出したレスポンスデータを受け取る．ハードウェア名から対応する公開鍵を取得し，ハードウェアの秘密鍵で暗号化されたレスポンスデータを復号化する．その結果を受付スクリプトが保存したチャレンジデータと比較し，一

致しない場合には認証失敗として扱う．認証に成功した場合，そのハードウェアが使用するプリンシパルの鍵情報 (keytab ファイル) を KDC から取得し，その内容をレスポンスとして返送する．

4.2 ユーザエントランスサーバ

ユーザエントランスサーバも Kerberos Keytab 配信サーバと同様に，認証処理を Apache 上の CGI で実装した．その構成も同様で，認証要求を受けチャレンジデータを送出する受付スクリプトと，クライアントからのレスポンスを受け検証・OS 情報の提供を行う応答スクリプトの 2 つで構成する．どちらも HTTPS サーバ認証セッション上でのやりとりを行うが，その際に Kerberos プリンシパルによる認証を必須とし，プリンシパルから認証要求を行ったハードウェアを識別できるようにしている．

このサーバでの処理は，HTTPS 上で Kerberos 認証を使う点，ハードウェアの秘密鍵ではなく利用者の認証デバイスの秘密鍵を使う点，認証成功時には keytab ファイルではなく起動する OS の情報が返される点を除き，Kerberos Keytab 配信サーバの処理と同様である．

4.3 クライアントシステム

クライアントシステムには，提案手法を適用するため，従来方式 [2] から次のような変更を行った．

（1） クライアントシステムのハードウェアの識別を X.509 から Kerberos ベースに変更

（2） ディスクイメージ全体のダウンロードから，Kerberized NFS によりディスクイメージが格納されているディレクトリをマウントする方式に変更

(1) は，クライアントシステムのハードウェアの識別方法の変更に伴うものであり，HTTPS SSL クライアント認証を使うか Kerberos プリンシパル認証を使うかの違いである．(2) は，ディスクイメージの扱いの変更に伴うものであり，HTTP クライアントを用いてディスクイメージを取得する処理が Kerberos 化された NFS サーバ上のディスクイメージにアクセスする準備を行う処理に置き換わる．

5. 評　　価

本研究で実装した提案方式の評価を行うため，認証処理に要する時間と利用者用 OS の起動に要する時間の計測を行った．

5.1 システム構成

図 5 に評価用に構成したシステムの構成を示す．Kerberos Keytab 配信サーバ，ユーザエントランスサーバ，バックエンドデータベース，Kerberos 化された NFS サーバは実験環境の都合から 1 台のサーバ上で実行する．このサーバと Kerberos KDC，クライアントシステムは 1000BASE-T のネットワークにそれぞれ接続されている．システムの各要素の構成は次の通りである．

- Kerberos Keytab 配信サーバ/ユーザエントランスサーバ/バックエンドデータベース/KerberizedNFS サーバ・クライアントシステム共通
  - OS: Linux 2.6.18(x86)

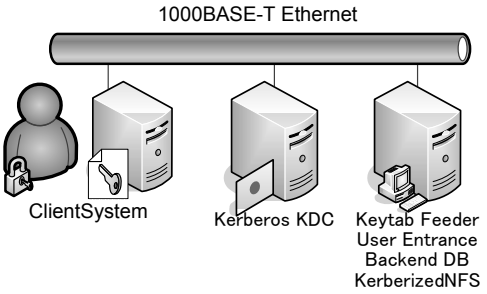

図 5　評価用システムの構成
Fig. 5　Components of evaluating system

  - CPU: Intel Core2Quad Q6600(2.4GHz)
  - メモリ: DDR2-SDRAM 4096MB
  - HDD: HGST SATA300 7200rpm
  - NIC: Intel PRO/1000 DesktopAdapter(1000BASE-T)
- Kerberos KDC
  - OS: Linux 2.6.18(x86)
  - CPU: AMD Athlon64 3500+ (2.2GHz)
  - メモリ: DDR-SDRAM 2048MB
  - HDD: HGST SATA300 7200rpm
  - NIC: Intel PRO/1000 DesktopAdapter(1000BASE-T)
- ネットワークスイッチ
  - DELL PowerConnect 2716(1000BASE-T)
- 認証デバイス (USB トークン)
  - 飛天ジャパン ePass2000
- サービス・デーモンほか
  - HTTP: Apache 2.2.3
  - HTTP Kerberos 認証: mod_spnego 0.1.0
  - Kerberos: MIT Kerberos 5
  - VMM: Xen 3.3.0
  - 言語: Ruby 1.8.5

5.2 実 験 方 法

提案手法による起動所要時間短縮化の効果を計測するため，認証所要時間と利用者用 OS の起動所要時間の計測を行う．

従来手法と提案手法の両方について，利用者用 OS を起動するための処理に必要な認証処理にかかる時間を計測し，これを認証所要時間とする．また，この認証処理が終了してから利用者用 OS が利用可能となるまでの時間を計測し，これを起動所要時間とする．

認証所要時間の計測にあたり使用する認証デバイスは，従来手法での計測を行った際に使用したものと同一のデバイスを用いる．計測する区間に関しても，従来手法での計測と同等となる区間を定め，この区間の実行にかかる時間を `time` コマンドにより記録する．その結果の平均値を計測結果として採用する．

起動所要時間に関しては，利用者用 OS としてどちらの手法でもディスクイメージサイズは 4GB である Windows XP SP3 を用いる．「利用可能」の判断のタイミングとしては，スタートアップに設定した Internet Explorer が，ホームページとして HDD 上にある画像の表示が完了した瞬間とする．

従来手法については，認証処理が終了し HTTPS でのダウン

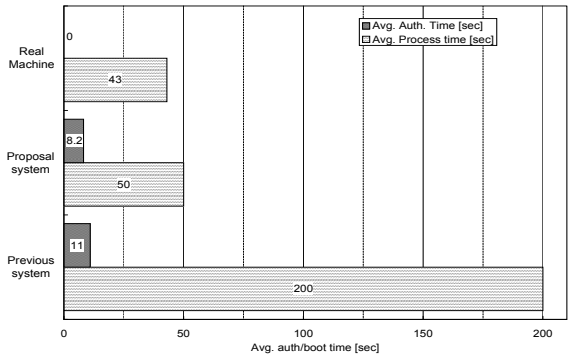

図 6　Windows XP 起動所要時間

Fig. 6　Time for booting Windows XP

ロードを開始してから Internet Explorer で画像が表示されるまでの時間を，提案手法ではディスクイメージにアクセスするために必要な処理(注1)を開始してから Internet Explorer で画像が表示されるまでの時間をそれぞれ計測する．なお，計測はストップウォッチによる目視確認により行う．この計測はそれぞれ 3 回行い，その平均値を計測結果として採用する．

5.3 実験結果

計測結果のグラフを図 6 に示す．認証所要時間については，従来手法での所要時間は平均 11 秒であるのに対し，提案手法では平均 8.2 秒であった．起動所要時間については，従来手法では平均 200 秒だったのに対し，提案手法では平均 50 秒となった．また，比較のために同じハードウェアの実機に直接 Windows XP をインストールし，ほぼ同等の構成を構築，起動時間を計測した．その結果，電源投入から Internet Explorer で画像が表示されるまで，平均 43 秒という結果であった．

5.4 考　　察

認証所要時間計測結果では，従来手法の平均 11 秒から平均 8.2 秒に短縮された．この違いは，使用するプロトコルの違い (従来手法では HTTPS+SSL クライアント認証，提案手法では HTTPS+Kerberos 認証) や，アクセス制御の実装の違いから現れたものだと考えられる．

起動所要時間計測結果では，従来手法の平均 200 秒から平均 50 秒に短縮された．OS イメージのサイズが 4GB と大きいため，ランダムアクセスによる短縮化の手法が有効であったことがわかる．実機に直接インストールした場合，約 43 秒で起動するのに対し，提案手法の場合では認証処理と起動処理をあわせて約 58 秒かかる．この差が提案手法のオーバーヘッドである．

ランダムアクセスによる起動所要時間の短縮化を行う上で，仮に起動所要時間が短縮できたとしてもそのために必要な認証にかかる時間が長くなってしまっては無意味である．実験結果より，この手法を適用する上で必要となる認証処理も含め，起動所要時間の短縮されたといえる．

提案手法を用いる場合，OS のディスクイメージへのアクセスが起こるたびにネットワークへのアクセスが生じる．従来手法では，ディスクイメージはローカルの HDD に保存されているため，OS 起動後のディスクアクセス速度の面では提案手法は従来手法に劣る．しかし，実験環境 (1000BASE-T) の場合では，Kerberos 化された NFS によるディスクアクセス速度は約 30MB/s であった．そのため，ディスクイメージ内の多量の小さなファイルにアクセスするようなケースや，大容量のデータにバーストアクセスするようなケースでなければ，体感速度の低下は目立たないと考えられる．また，そういった用途での使用の際には，Unionfs [7] のような仕組みを利用することにより，必要に応じてローカルの HDD を利用できるようにすれば，問題を解決できると考えられる．

(注1)：rpc.gssd デーモンの起動やチケットの取得，mount コマンドの実行

6. おわりに

本稿では，認証デバイスを用いた OS の起動・終了管理システムにおいてランダムアクセス可能なネットワークファイルシステムを用いることによりその起動に要する時間を短縮する手法について述べた．そこで用いるファイルシステムとしては，1) アクセス制御の仕組みがあり，2) ランダムアクセス可能であること，という条件が必要であることを述べ，これを満たすものとして Kerberos 化された NFS が適切であることを述べた．

この手法の有効性を確認するため，OS 制御レイヤーのシステムと提案手法に必要となる各種サーバを構築し，従来手法との比較を行った．比較の観点としては，起動所要時間と認証所要時間を採用した．その結果，起動所要時間の短縮化が実現されたことを確認し，またこの手法を採用するために加えた認証処理への変更も，短縮化の結果を打ち消すような悪化は起こさないことを確認した．

今後の課題としては，利用者用 OS の稼働中に認証デバイスが抜かれた場合に，利用者用 OS をシャットダウンさせるのではなく，適宜ゲスト OS を中断させる仕組みの実現や，アクセス制御の規模が拡大した場合の動作検証や改善案の検討を行う．ローカルの HDD をキャッシュとして利用する．また，本稿では実験のためにアクセス制御リストをファイル名ベースの単純な実装で提供したが，大規模な運用にも耐えられるよう，データベースを利用した仕組みも実装する．

文　　献

[1] 高田真吾, 佐藤聡, 新城靖, 中井央, 板野肯三: “認証デバイスを用いた OS の安全な起動制御”, 情報処理学会 研究報告 2008-IOT-1, pp. 77–82 (2008).

[2] 高田真吾, 佐藤聡, 新城靖, 中井央, 板野肯三 : “認証デバイスを用いた OS の起動・終了制御”, 情報処理学会論文誌, **Vol. 50**, No. 3 (2009). (印刷中).

[3] K. Suzaki, T. Yagi, K. Iijima, et al.: “Trusted boot of http-fuse knoppix”, Linux-Kongress 2006 (2006).

[4] S. Bajikar: “Trusted platform module(tpm) based security on notebook pcs-white paper *http://www.intel.com/design/mobile/platform/downloads/Trusted_Platform_Module_White_Paper.pdf*” (2002).

[5] R. A. Haynes: “Nfs, kerberos, and unicos”, Cray users group conference, pp. 23–27 (1991).

[6] P. Barham, B. Dragovic, K. Fraser, et al.: “Xen and the art of virtualization”, SIGOPS Oper. Syst. Rev., **Vol. 37**, No. 5, pp. 164–177 (2003).

[7] D. P. Quigley, J. Sipek, C. P. Wright and E. Zadok: “UnionFS: User- and Community-oriented Development of a Unification Filesystem”, Proceedings of the 2006 Linux Symposium, pp. 349–362 (2006).